An Introduction to Stata Programming

Second Edition
(Pages omitted)
# Contents

- List of figures xvii
- List of tables xix
- Preface xxii
- Acknowledgments xxiii
- Notation and typography xxv

1 Why should you become a Stata programmer? 1
   - Do-file programming 2
   - Ado-file programming 2
   - Mata programming for ado-files 2
   1.1 Plan of the book 3
   1.2 Installing the necessary software 4

2 Some elementary concepts and tools 5
   2.1 Introduction 5
      - What you should learn from this chapter 5
   2.2 Navigational and organizational issues 5
      - The current working directory and profile.do 6
      - Locating important directories: sysdir and adopath 6
      - Organization of do-files, ado-files, and data files 6
   2.3 Editing Stata do- and ado-files 6
   2.4 Data types 6
      - Storing data efficiently: The compress command 11
      - Date and time handling 11
      - Time-series operators 13
      - Factor variables and operators 14
2.5 Handling errors: The capture command ................................... 16
2.6 Protecting the data in memory: The preserve and restore commands ... 17
2.7 Getting your data into Stata .................................................... 18
  2.7.1 Inputting and importing data ............................................ 18
    Handling text files ......................................................... 18
    Free format versus fixed format ........................................ 20
    The import delimited command .......................................... 21
    Accessing data stored in spreadsheets ............................... 23
    Fixed-format data files .................................................. 24
  2.7.2 Importing data from other package formats .......................... 29
2.8 Guidelines for Stata do-file programming style ......................... 30
  2.8.1 Basic guidelines for do-file writers ................................ 31
  2.8.2 Enhancing speed and efficiency ..................................... 33
2.9 How to seek help for Stata programming ................................ 33

3 Do-file programming: Functions, macros, scalars, and matrices ......... 37
  3.1 Introduction ........................................................................ 37
    3.1.1 What you should learn from this chapter ...................... 37
  3.2 Some general programming details ...................................... 38
    3.2.1 The varlist .............................................................. 39
    3.2.2 The numlist ............................................................ 39
    3.2.3 The if exp and in range qualifiers ............................. 39
    3.2.4 Missing-data handling .............................................. 40
      Recoding missing values: The mvdecode and mvencode commands ........................................... 41
    3.2.5 String-to-numeric conversion and vice versa ................ 42
      Numeric-to-string conversion ........................................ 43
      Working with quoted strings ........................................ 44
  3.3 Functions for the generate command .................................. 44
    3.3.1 Using if exp with indicator variables ........................ 45
    3.3.2 The cond() function .............................................. 46
## Contents

3.3.3 Recoding discrete and continuous variables ................. 49
3.4 Functions for the egen command ................................. 51
   Official egen functions ..................................... 52
       egen functions from the user community .................. 53
3.5 Computation for by-groups ........................................ 54
   3.5.1 Observation numbering: _n and _N .......................... 55
3.6 Local macros ..................................................... 55
3.7 Global macros .................................................... 60
3.8 Extended macro functions and macro list functions ............ 60
   3.8.1 System parameters, settings, and constants: creturn ...... 62
3.9 Scalars ............................................................ 62
3.10 Matrices .......................................................... 64

4 Cookbook: Do-file programming I ................................. 67
   4.1 Tabulating a logical condition across a set of variables ....... 67
   4.2 Computing summary statistics over groups .................... 69
   4.3 Computing the extreme values of a sequence .................. 70
   4.4 Computing the length of spells .............................. 71
   4.5 Summarizing group characteristics over observations ......... 76
   4.6 Using global macros to set up your environment .............. 78
   4.7 List manipulation with extended macro functions ............ 79
   4.8 Using creturn values to document your work .................. 81

5 Do-file programming: Validation, results, and data management 83
   5.1 Introduction ....................................................... 83
       5.1.1 What you should learn from this chapter ................ 83
   5.2 Data validation: The assert, count, and duplicates commands .... 83
   5.3 Reusing computed results: The return and ereturn commands .... 90
       5.3.1 The ereturn list command ............................... 94
   5.4 Storing, saving, and using estimated results ................... 97
       5.4.1 Generating publication-quality tables from stored estimates 102
   5.5 Reorganizing datasets with the reshape command ............... 104
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>Combining datasets</td>
<td>109</td>
</tr>
<tr>
<td>5.7</td>
<td>Combining datasets with the append command</td>
<td>111</td>
</tr>
<tr>
<td>5.8</td>
<td>Combining datasets with the merge command</td>
<td>113</td>
</tr>
<tr>
<td>5.8.1</td>
<td>The one-to-one match-merge</td>
<td>115</td>
</tr>
<tr>
<td>5.8.2</td>
<td>The dangers of many-to-many merges</td>
<td>116</td>
</tr>
<tr>
<td>5.9</td>
<td>Other data management commands</td>
<td>117</td>
</tr>
<tr>
<td>5.9.1</td>
<td>The fillin command</td>
<td>117</td>
</tr>
<tr>
<td>5.9.2</td>
<td>The cross command</td>
<td>117</td>
</tr>
<tr>
<td>5.9.3</td>
<td>The stack command</td>
<td>118</td>
</tr>
<tr>
<td>5.9.4</td>
<td>The separate command</td>
<td>119</td>
</tr>
<tr>
<td>5.9.5</td>
<td>The joinby command</td>
<td>120</td>
</tr>
<tr>
<td>5.9.6</td>
<td>The xpose command</td>
<td>121</td>
</tr>
<tr>
<td>6</td>
<td>Cookbook: Do-file programming II</td>
<td>123</td>
</tr>
<tr>
<td>6.1</td>
<td>Efficiently defining group characteristics and subsets</td>
<td>123</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Using a complicated criterion to select a subset of observations</td>
<td>124</td>
</tr>
<tr>
<td>6.2</td>
<td>Applying reshape repeatedly</td>
<td>125</td>
</tr>
<tr>
<td>6.3</td>
<td>Handling time-series data effectively</td>
<td>129</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Working with a business-daily calendar</td>
<td>132</td>
</tr>
<tr>
<td>6.4</td>
<td>reshape to perform rowwise computation</td>
<td>133</td>
</tr>
<tr>
<td>6.5</td>
<td>Adding computed statistics to presentation-quality tables</td>
<td>136</td>
</tr>
<tr>
<td>6.6</td>
<td>Presenting marginal effects rather than coefficients</td>
<td>138</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Graphing marginal effects with marginsplot</td>
<td>140</td>
</tr>
<tr>
<td>6.7</td>
<td>Generating time-series data at a lower frequency</td>
<td>141</td>
</tr>
<tr>
<td>6.8</td>
<td>Using suest and gsem to compare estimates from nonoverlapping samples</td>
<td>146</td>
</tr>
<tr>
<td>6.9</td>
<td>Using reshape to produce forecasts from a VAR or VECM</td>
<td>149</td>
</tr>
<tr>
<td>6.10</td>
<td>Working with IRF files</td>
<td>152</td>
</tr>
<tr>
<td>7</td>
<td>Do-file programming: Prefixes, loops, and lists</td>
<td>157</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>157</td>
</tr>
<tr>
<td>7.1.1</td>
<td>What you should learn from this chapter</td>
<td>157</td>
</tr>
</tbody>
</table>
Contents

7.2 Prefix commands ........................................... 157
  7.2.1 The by prefix ........................................... 158
  7.2.2 The statsby prefix ..................................... 160
  7.2.3 The xi prefix and factor-variable notation ............... 161
  7.2.4 The rolling prefix ..................................... 162
  7.2.5 The simulate and permute prefixes ....................... 164
  7.2.6 The bootstrap and jackknife prefixes ..................... 167
  7.2.7 Other prefix commands ................................ 169

7.3 The forvalues and foreach commands ....................... 169

8 Cookbook: Do-file programming III .......................... 177
  8.1 Handling parallel lists .................................... 177
  8.2 Calculating moving-window summary statistics ............... 178
    8.2.1 Producing summary statistics with rolling and merge .... 180
    8.2.2 Calculating moving-window correlations ................ 181
  8.3 Computing monthly statistics from daily data ............... 182
  8.4 Requiring at least n observations per panel unit ............ 184
  8.5 Counting the number of distinct values per individual ....... 185
  8.6 Importing multiple spreadsheet pages ...................... 186

9 Do-file programming: Other topics ....................... 189
  9.1 Introduction ........................................... 189
    9.1.1 What you should learn from this chapter ............... 189
  9.2 Storing results in Stata matrices ........................ 189
  9.3 The post and postfile commands .......................... 193
  9.4 Output: The export delimited, outfile, and file commands ... 196
  9.5 Automating estimation output ............................ 199
  9.6 Automating graphics ..................................... 203
  9.7 Characteristics ......................................... 207

10 Cookbook: Do-file programming IV .......................... 211
  10.1 Computing firm-level correlations with multiple indices .... 211
  10.2 Computing marginal effects for graphical presentation .... 214
10.3 Automating the production of \LaTeX\ tables \hspace{1cm} 216
10.4 Extracting data from graph files' sersets \hspace{1cm} 220
10.5 Constructing continuous price and returns series \hspace{1cm} 223

11 Ado-file programming \hspace{1cm} 231
11.1 Introduction \hspace{1cm} 231
\hspace{1cm} 11.1.1 What you should learn from this chapter \hspace{1cm} 232
11.2 The structure of a Stata program \hspace{1cm} 233
11.3 The program statement \hspace{1cm} 233
11.4 The syntax and return statements \hspace{1cm} 234
11.5 Implementing program options \hspace{1cm} 237
11.6 Including a subset of observations \hspace{1cm} 238
11.7 Generalizing the command to handle multiple variables \hspace{1cm} 240
11.8 Making commands byable \hspace{1cm} 242
\hspace{1cm} Program properties \hspace{1cm} 243
11.9 Documenting your program \hspace{1cm} 244
11.10 egen function programs \hspace{1cm} 246
11.11 Writing an e-class program \hspace{1cm} 248
\hspace{1cm} 11.11.1 Defining subprograms \hspace{1cm} 250
11.12 Certifying your program \hspace{1cm} 250
11.13 Programs for ml, nl, and nlsur \hspace{1cm} 252
\hspace{1cm} Maximum likelihood estimation of distributions' parameters \hspace{1cm} 253
\hspace{1cm} 11.13.1 Writing an ml-based command \hspace{1cm} 260
\hspace{1cm} 11.13.2 Programs for the nl and nlsur commands \hspace{1cm} 263
11.14 Programs for gmm \hspace{1cm} 265
11.15 Programs for the simulate, bootstrap, and jackknife prefixes \hspace{1cm} 270
11.16 Guidelines for Stata ado-file programming style \hspace{1cm} 272
\hspace{1cm} 11.16.1 Presentation \hspace{1cm} 273
\hspace{1cm} 11.16.2 Helpful Stata features \hspace{1cm} 274
\hspace{1cm} 11.16.3 Respect for datasets \hspace{1cm} 274
\hspace{1cm} 11.16.4 Speed and efficiency \hspace{1cm} 275
11.16.5 Reminders .............................................. 275
11.16.6 Style in the large ................................. 276
11.16.7 Use the best tools .............................. 276

12 Cookbook: Ado-file programming 277
12.1 Retrieving results from rolling .................. 277
12.2 Generalization of egen function pct9010() to support all pairs of quantiles . . . . . . . . . 280
12.3 Constructing a certification script ............... 282
12.4 Using the ml command to estimate means and variances .......... 287
  12.4.1 Applying equality constraints in ml estimation ............ 289
12.5 Applying inequality constraints in ml estimation ............ 291
12.6 Generating a dataset containing the longest spell .......... 294
12.7 Using suest on a fixed-effects model ............. 297

13 Mata functions for do-file and ado-file programming 301
13.1 Mata: First principles ................................. 301
  13.1.1 What you should learn from this chapter .......... 302
13.2 Mata fundamentals ..................................... 302
  13.2.1 Operators ............................................. 303
  13.2.2 Relational and logical operators .................. 304
  13.2.3 Subscripts ............................................. 305
  13.2.4 Populating matrix elements ....................... 305
  13.2.5 Mata loop commands ............................... 307
  13.2.6 Conditional statements ............................ 308
13.3 Mata’s st_ interface functions .................... 309
  13.3.1 Data access .......................................... 309
  13.3.2 Access to locals, globals, scalars, and matrices ....... 311
  13.3.3 Access to Stata variables’ attributes .............. 312
13.4 Calling Mata with a single command line .......... 312
13.5 Components of a Mata function .................. 316
  13.5.1 Arguments ............................................. 316
Contents

13.5.2 Variables .................................................. 315
13.5.3 Stored results ............................................. 315
13.6 Calling Mata functions ....................................... 318
13.7 Example: stinterface function usage ...................... 320
13.8 Example: Matrix operations ................................. 322
  13.8.1 Extending the command ................................. 325
13.9 Mata-based likelihood function evaluators .................. 326
13.10 Creating arrays of temporary objects with pointers ......... 331
13.11 Structures .................................................. 334
13.12 Additional Mata features .................................... 337
  13.12.1 Macros in Mata functions ............................... 337
  13.12.2 Associative arrays in Mata functions ................... 338
  13.12.3 Compiling Mata functions ............................... 340
  13.12.4 Building and maintaining an object library ............ 341
  13.12.5 A useful collection of Mata routines ................... 342
14 Cookbook: Mata function programming ......................... 343
  14.1 Reversing the rows or columns of a Stata matrix ......... 343
  14.2 Shuffling the elements of a string variable .............. 346
  14.3 Firm-level correlations with multiple indices with Mata .. 348
  14.4 Passing a function to a Mata function ..................... 353
  14.5 Using subviews in Mata ................................... 356
  14.6 Storing and retrieving country-level data with Mata .. 358
  14.7 Locating nearest neighbors with Mata ..................... 363
  14.8 Using a permutation vector to reorder results ............ 368
  14.9 Producing \LaTeX\ tables from svy results ................. 370
  14.10 Computing marginal effects for quantile regression ...... 375
  14.11 Computing the seemingly unrelated regression estimator .. 379
  14.12 A GMM-CUE estimator using Mata’s optimize() functions .. 384
Contents

References 397
Author index 403
Subject index 405
(Pages omitted)
Preface

This book is a concise introduction to the art of Stata programming. It covers three types of programming that can be used in working with Stata: do-file programming, ado-file programming, and Mata functions that work in conjunction with do- and ado-files. Its emphasis is on the automation of your work with Stata and how programming on one or more of these levels can help you use Stata more effectively.

In the development of these concepts, I do not assume that you have prior experience with Stata programming, although familiarity with the command-line interface is helpful. While examples are drawn from several disciplines, my background as an applied econometrician is evident in the selection of some sample problems. The introductory first chapter motivates the why: why should you invest time and effort into learning Stata programming? In chapter 2, I discuss elementary concepts of the command-line interface and describe some commonly used tools for working with programs and datasets.

The format of the book may be unfamiliar to readers who have some familiarity with other books that help you learn how to use Stata. Beginning with chapter 4, each even-numbered chapter is a “cookbook” chapter containing several “recipes”, 47 in total. Each recipe poses a problem: how can I perform a certain task with Stata programming? The recipe then provides a complete worked solution to the problem and describes how the features presented in the previous chapter can be put to good use. You may not want to follow a recipe exactly from the cookbook; just as in cuisine, a minor variation on the recipe may meet your needs, or the techniques presented in that recipe may help you see how Stata programming applies to your specific problem.

Most Stata users who delve into programming use do-files to automate and document their work. Consequently, the major focus of the book is do-file programming, covered in chapters 3, 5, 7, and 9. Some users will find that writing formal Stata programs, or ado-files, meets their needs. Chapter 11 is a concise summary of ado-file programming, with the cookbook chapter that follows presenting several recipes that contain developed ado-files. Stata’s matrix programming language, Mata, can also be helpful in automating certain tasks. Chapter 13 presents a summary of Mata concepts and the key features that allow interchange of variables, scalars, macros, and matrices. The last chapter, cookbook chapter 14, presents several examples of Mata functions developed to work with ado-files. All the do-files, ado-files, Mata functions, and datasets used in the book’s examples and recipes are available from the Stata Press website, as discussed in Notation and typography.
The second edition of this book contains several new recipes illustrating how do-files, ado-files, and Mata functions can be used to solve programming problems. Several recipes have also been updated to reflect new features in Stata added between versions 10 and 14. The discussion of maximum-likelihood function evaluators has been significantly expanded in this edition. The new topics covered in this edition include factor variables and operators; use of \texttt{margins}, \texttt{marginsplot}, and \texttt{suest}; Mata-based likelihood function evaluators; and associative arrays.
(Pages omitted)
1 Why should you become a Stata programmer?

This book provides an introduction to several contexts of Stata programming. I must first define what I mean by “programming”. You can consider yourself a Stata programmer if you write do-files, which are text files of sequences of Stata commands that you can execute with the `do` ([R] do) command, by double-clicking on the file, or by running them in the Do-file Editor ([R] doedit). You might also write what Stata formally defines as a program, which is a set of Stata commands that includes the `program` ([P] program) command. A Stata program, stored in an ado-file, defines a new Stata command. You can also use Stata’s matrix programming language, Mata, to write routines in that language that are called by ado-files. Any of these tasks involves Stata programming.

With that set of definitions in mind, we must deal with the why: why should you become a Stata programmer? After answering that essential question, this text takes up the how: how you can become a more efficient user of Stata by using programming techniques, be they simple or complex.

Using any computer program or language is all about efficiency—getting the computer to do the work that can be routinely automated, reducing human errors, and allowing you to more efficiently use your time. Computers are excellent at performing repetitive tasks; humans are not. One of the strongest rationales for learning how to use programming techniques in Stata is the potential to shift more of the repetitive burden of data management, statistical analysis, and production of graphics to the computer. Let’s consider several specific advantages of using Stata programming techniques in the three contexts listed above.

1. There are also specialized forms of Stata programming, such as dialog programming, scheme programming, and class programming. A user-written program can present a dialog, like any official Stata command, if its author writes a dialog file. The command can also be added to the User menu of Stata’s graphical interface. For more information, see [P] dialog programming and [P] window programming. Graphics users can write their own schemes to set graphic defaults. See [G-4] schemes intro for details. Class programming allows you to write object-oriented programs in Stata. As [P] class indicates, this has primarily been used in Stata’s graphics subsystem and graphical user interface. I do not consider these specialized forms of programming in this book.
Chapter 1  Why should you become a Stata programmer?

Do-file programming

Using a do-file to automate a specific data-management or statistical task leads to reproducible research and the ability to document the empirical research process. This reduces the effort needed to perform a similar task at a later point or to document for your coworkers or supervisor the specific steps you followed. Ideally, your entire research project should be defined by a set of do-files that execute every step, from the input of the raw data to the production of the final tables and graphs. Because a do-file can call another do-file (and so on), a hierarchy of do-files can be used to handle a complex project.

The beauty of this approach is its flexibility. If you find an error in an earlier stage of the project, you need only to modify the code and then rerun that do-file and those following to bring the project up to date. For instance, a researcher may need to respond to a review of her paper—submitted months ago to an academic journal—by revising the specification of variables in a set of estimated models and estimating new statistical results. If all the steps that produce the final results are documented by a set of do-files, her task is straightforward. I argue that all serious users of Stata should gain some facility with do-files and the Stata commands that support repetitive use of commands.

That advice does not imply that Stata’s interactive capabilities should be shunned. Stata is a powerful and effective tool for exploratory data analysis and ad hoc queries about your data. But data-management tasks and the statistical analyses leading to tabulated results should not be performed with “point-and-click” tools that leave you without an audit trail of the steps you have taken.

Ado-file programming

On a second level, you may find that despite the breadth of Stata’s official and user-written commands, there are tasks you must repeatedly perform that involve variations on the same do-file. You would like Stata to have a command to perform those tasks. At that point, you should consider Stata’s ado-file programming capabilities. Stata has great flexibility: a Stata command need be no more than a few lines of Stata code. Once defined, that command becomes a “first-class citizen”. You can easily write a Stata program, stored in an ado-file, that handles all the features of official Stata commands such as `if exp, in range`, and command options. You can (and should) write a help file that documents the program’s operation for your benefit and for those with whom you share the code. Although ado-file programming requires that you learn how to use some additional commands used in that context, it can help you become more efficient in performing the data-management, statistical, or graphical tasks that you face.

Mata programming for ado-files

On a third level, your ado-files can perform some complicated tasks that involve many invocations of the same commands. Stata’s ado-file language is easy to read and write,
1.1 Plan of the book

The chapters of this book present the details of the three types of Stata programming discussed above, placing the greatest emphasis on effective use of do-file programming. Each fairly brief chapter on the structure of programming techniques is followed by a “cookbook” chapter. These chapters contain several “recipes” for the solution of a particular, commonly encountered problem, illustrating the necessary programming techniques to compose a solution. Like in a literal cookbook, the recipes here are illustrative examples; you are free to modify the ingredients to produce a somewhat different dish. The recipes as presented may not address your precise problem, but they should prove helpful in devising a solution as a variation on the same theme.

---

2. For details on the SSC (Boston College) archive of user-contributed routines, type `help ssc`. 
(Pages omitted)
4 Cookbook: Do-file programming I

This cookbook chapter presents for Stata do-file programmers several recipes using the programming features described in the previous chapter. Each recipe poses a problem and a worked solution. Although you may not encounter this precise problem, you should be able to recognize its similarities to a task that you would like to automate in a do-file.

4.1 Tabulating a logical condition across a set of variables

The problem.

When considering many related variables, you want to determine whether, for each observation, all variables satisfy a logical condition. Alternatively, you might want to know whether any satisfy that condition (for instance, taking on inappropriate values), or you might want to count how many of the variables satisfy the logical condition.[1]

The solution.

This would seem to be a natural application of `egen` ([P] egen), because that command already contains many rowwise functions to perform computations across variables. For instance, the `anycount()` function counts the number of variables in its varlist whose values for each observation match those of an integer numlist, whereas the `rowmiss()` and `rownonmiss()` functions tabulate the number of missing and nonmissing values for each observation, respectively. The three tasks above are all satisfied by `egen` functions from Nicholas Cox’s `egenmore` package: `rall()`, `rany()`, and `rcount()`, respectively. Why not use those functions, then?

Two reasons come to mind: First, recall that `egen` functions are interpreted code. Unlike the built-in functions accessed by `generate`, the logic of an `egen` function must be interpreted each time it is called. For a large dataset, the time penalty can be significant. Second, to use an `egen` function, you must remember that there is such a function, and you must remember its name. In addition to Stata’s official `egen` functions, documented in the online help files, there are many user-written `egen` functions available, but you must track them down.

For these reasons, current good programming practice suggests that you should avoid `egen` function calls in instances where the performance penalty might be an issue. This

[1] This recipe relies heavily on Nicholas J. Cox’s `egenmore` help file.